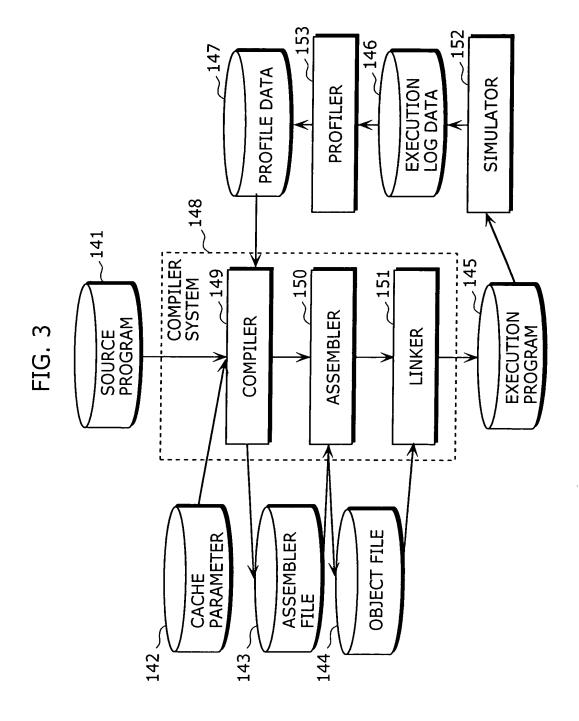
FIG. 1

```
int a[];
for (i=0; i<128; i++){
    dpref(&a[i + N]); // Prefetch data a certain number (N, in this case) of iterations ahead
    x += a[i]; // in consideration of latency caused until reference is made
   int a[];
for (i=0; i<128; i++){
    x += a[i];
                                                                                                                                                                                                                                                                          (p)
(a)
```

FIG. 2



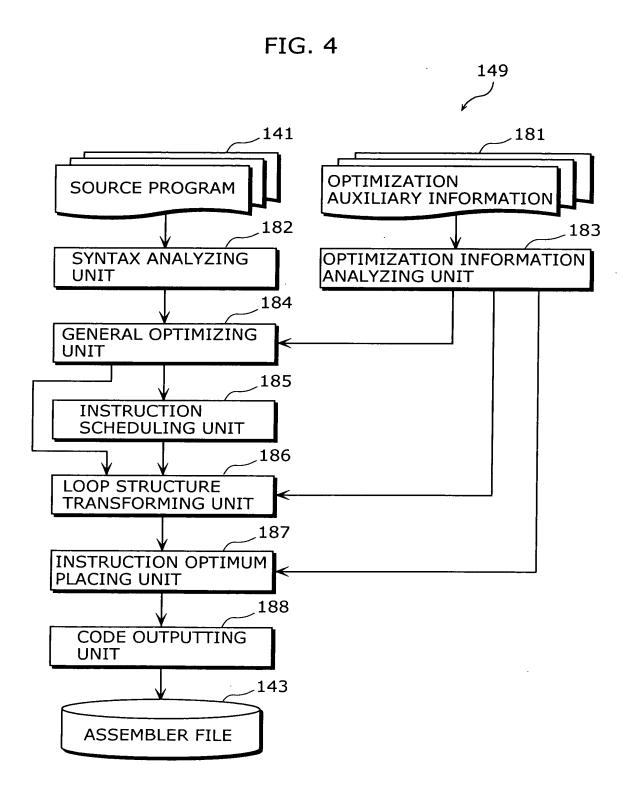
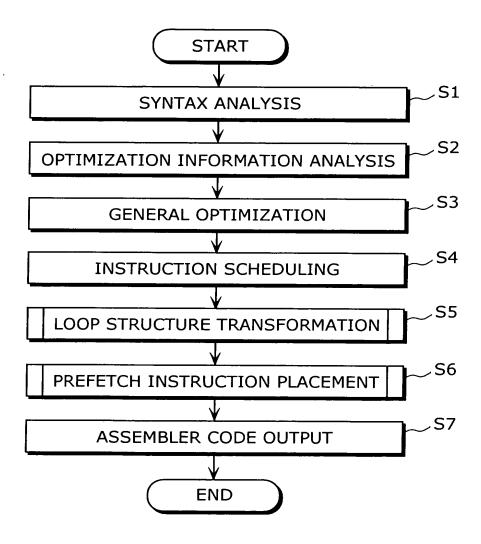


FIG. 5



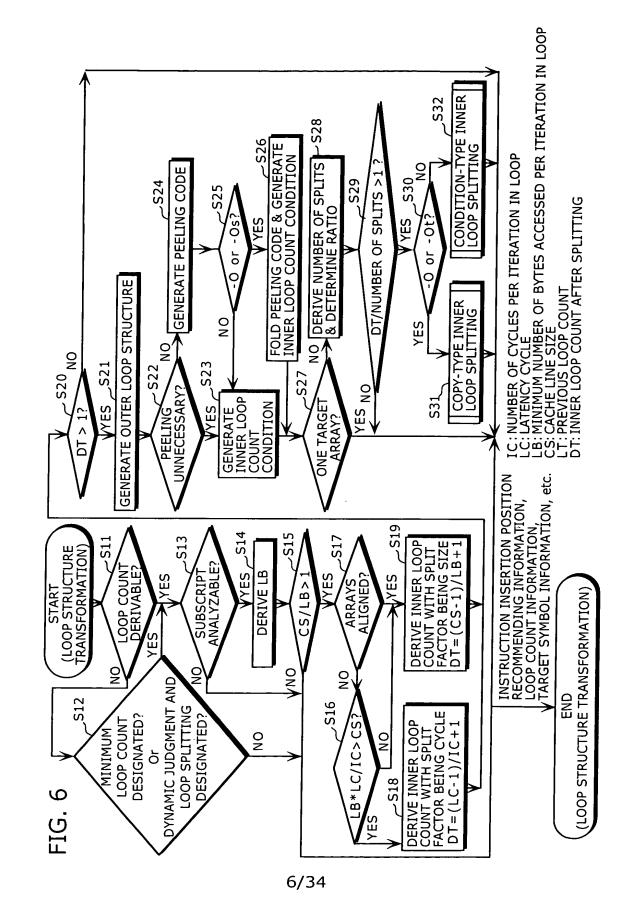


FIG. 7

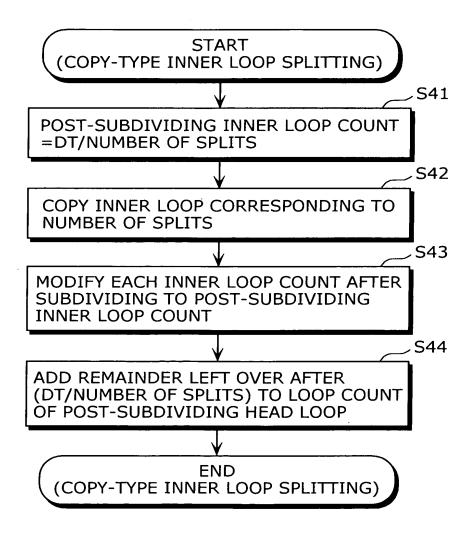
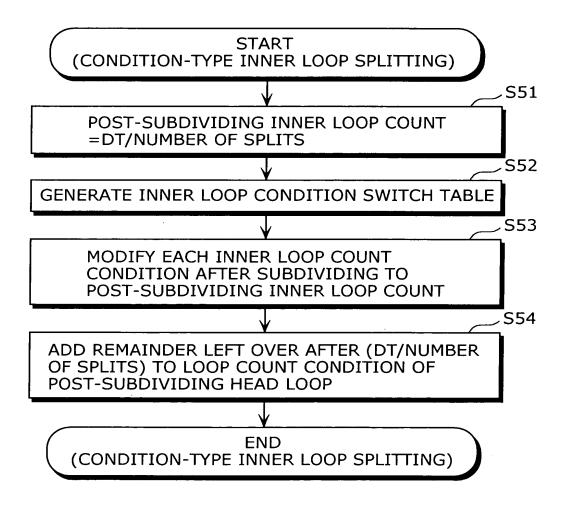


FIG. 8



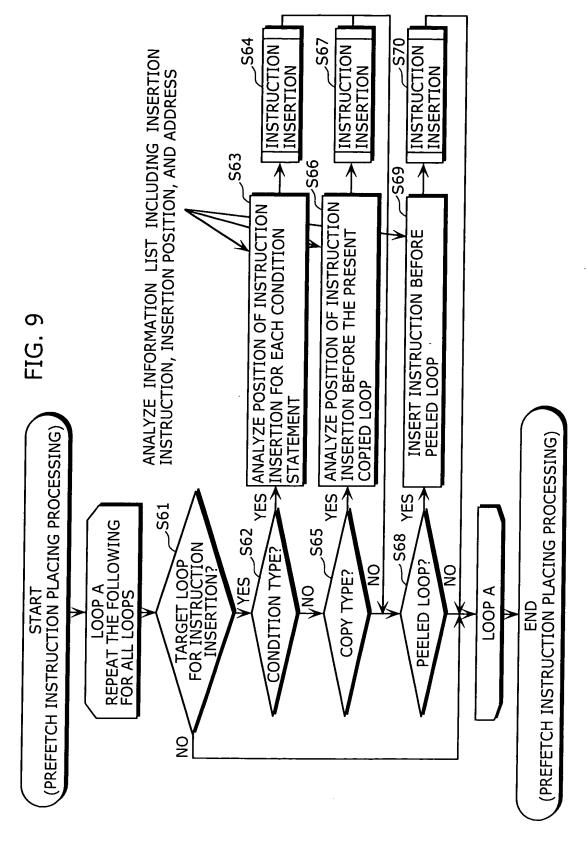


FIG. 10

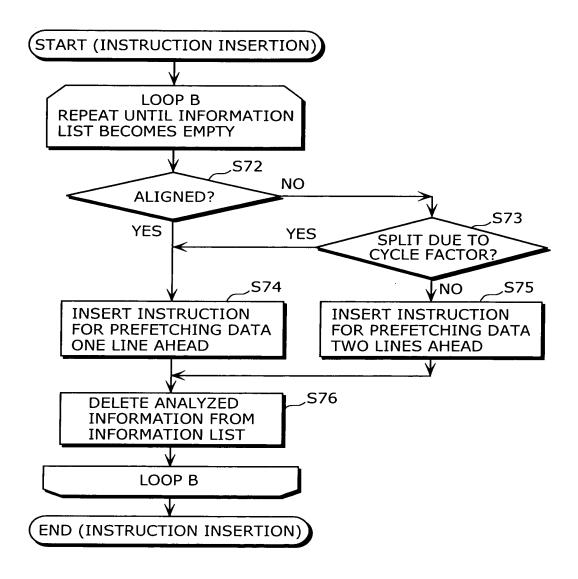


FIG. 11

```
PEELING IS UNNECESSARY
(a) A IS OF FOUR-BYTE TYPE.
                               282
    for(i=0;i<128;i++){
       sum+=A[i];
    }
        STRUCTURE
        TRANSFORMATION |
(b)
                               284
    for(i=0;i<128;){
      for(j=0;j<32;j++,i++){}
        sum+=A[i];
      }
    }
          INSTRUCTION
          INSERTION
                              286
(c)
   for(i=0;i<128;){
      dpref(&A[i+32]);
      for(j=0;j<32;j++,i++){
          sum+=A[i];
     }
   }
```

FIG. 12

INPUT PROGRAM SOURCE IN C LANGUAGE

```
int A[1000];
int main(void)
{
    int i:
    int sum = 0;

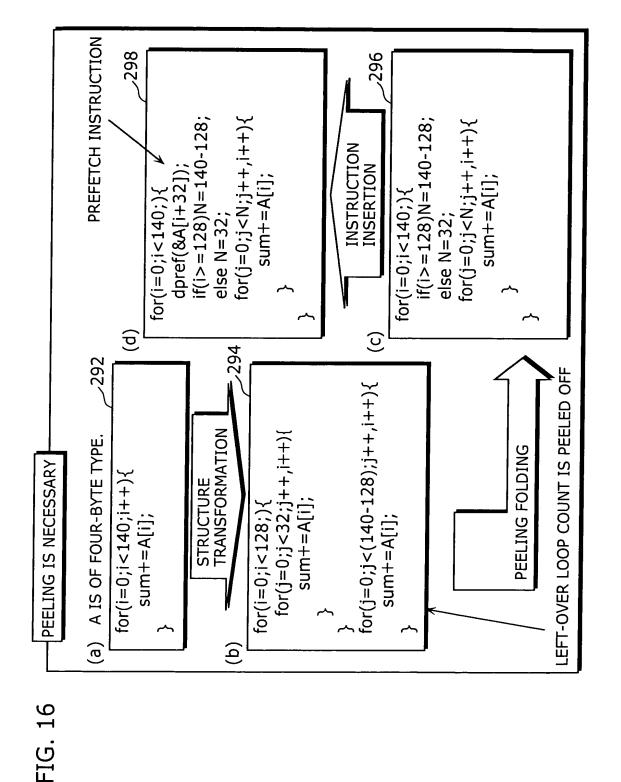
    for ( i=0; j<128;i++) {
        sum += A[ i ];
    }

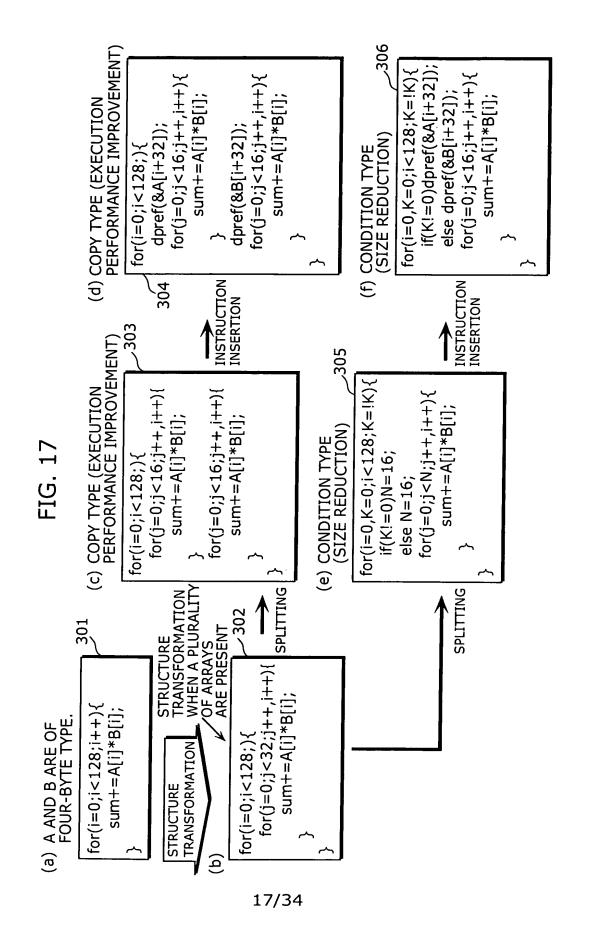
    return sum;
}</pre>
```

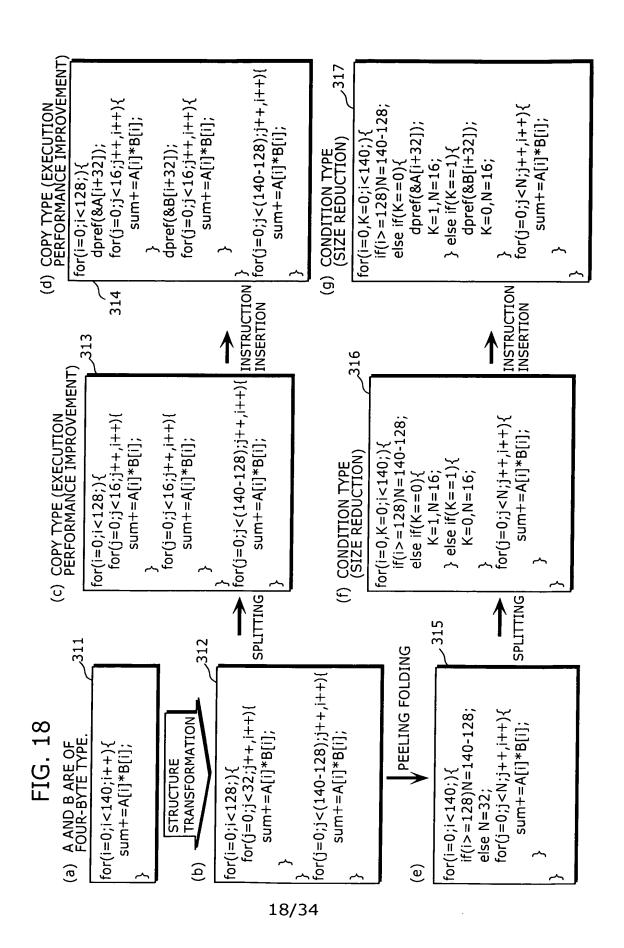
FIG. 13 🗓

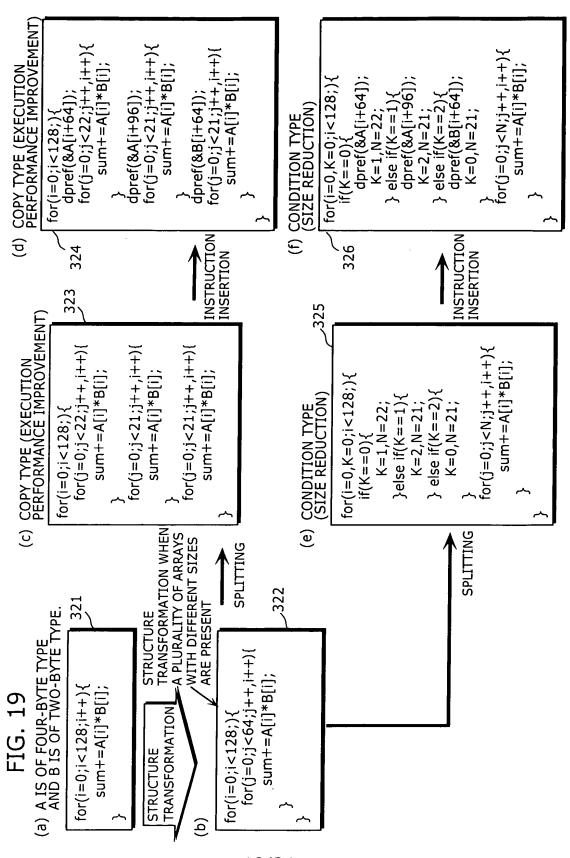
\sim	INTERMEDIATE LAN	INTERMEDIATE LANGUAGE INPUT BY TRANSFORMING UNIT	DRMING UNIT
	[PROLOG] [BGNBBLK] B1	[predecess set] no	[success set] B2
	wom.		7
	ک سے ا	 -	
	у М М	REG (vr3) IMM(_A\$) REG (vr1) IMM(vr2)	
	[ENDBBLK]		
	[BGNBBLK] B2	[predecess set] B1 B2	[success set] B2 B3
	add	REG (vr2) REG (vr2), I REG (vr4), REG(vr3) IN	MM(1)
	cmplt add	FLAG(C6) REG (vr2), REG REG (vr1) REG (vr4). REG	FLAG(C6) REG (vr2), REG (vr5) REG (vr1) REG (vr4), REG (vr1)
	jmpf [ENDBBLK]	FLÀG (C6), LAB(L0000	1)
	[BGNBBLK] B3	[predecess set] B2	[success set] no
	ret	NEG (10) NEG (NII)	
	[ENDBBLK] [EPILOG]		

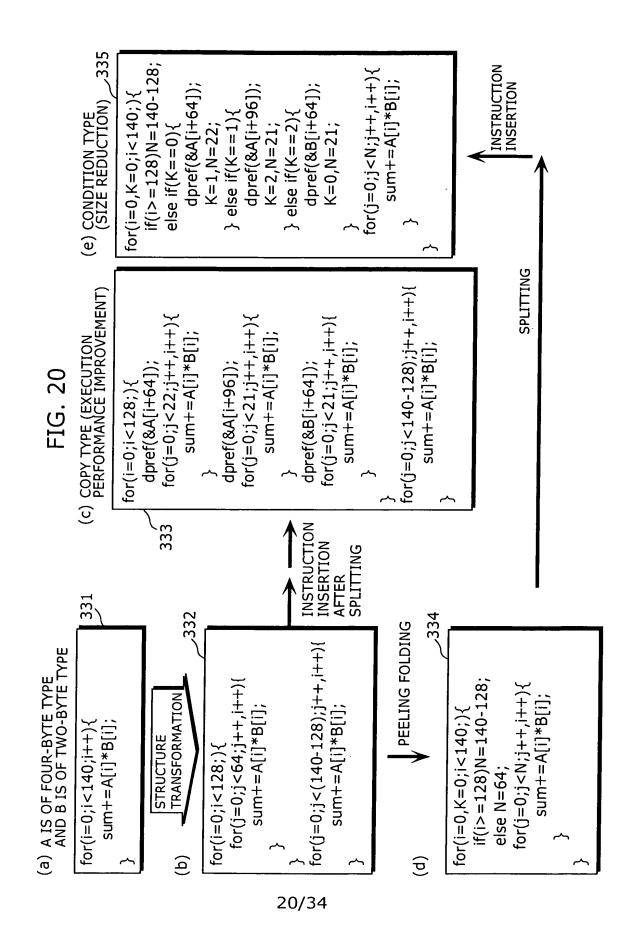
FIG. 15		INTERMEDIATE LANGUAGE AFTER INSTRUCTION INSERTION	TRUCTION	INSERTION 270
	[PROLOG] [BGNBBLK] B1	[predecess set] no	01	[success set] B4
			REG (vr2)	[IMM(0)
		^0	(Vr5)	IMM(32)
			REG (vr3)	MEM(_A\$)
			(Vr1)	REG(vr2)
	ראוממהאבז	mov	REG (vr7)	IMM(128)
	[BGNBBLK] B4	[predecess set] B1 B4	B1 B4	[success set] B2
	[FNDRRIK]	dpref	INDIREC	INDIRECT(vr2, 128), REG (vr2)
	[BGNBBLK] B2 [label] L00001	[predecess set] B4 B2	B4 B2	[success set] B2 B5
		add R Idinc R	(EG (vr2) (EG (vr4), f	REG (vr2) REG (vr2), IMM(1) REG (vr4), REG(vr3) INDIRECT(vr3 0) REG(vr3) IMM(1)
			LAG(C6)	REG (vr2), REG (vr5)
		add impf	(EG (vr1) Fl AG (CA	REG (vr4), REG (vr1)
	[ENDBBLK]			// LAB(LUUUUI.)
	[BGNBBLK] B5	redecess set	B2	[success set] B5 B3
		cmplt impf	LAG (C6) FI AG (C6	FLAG (C6) REG (vr2), REG (vr7)
	[ENDBBLK]), LAB(LUUUUZ)
	[BGNBBLK] B3	[predecess set] B5 mov REG] B5 [success] REG (r0) REG (vr1)	[success set] no REG (vr1)
		ret		
	[ENDBBLK] [EPILOG]			

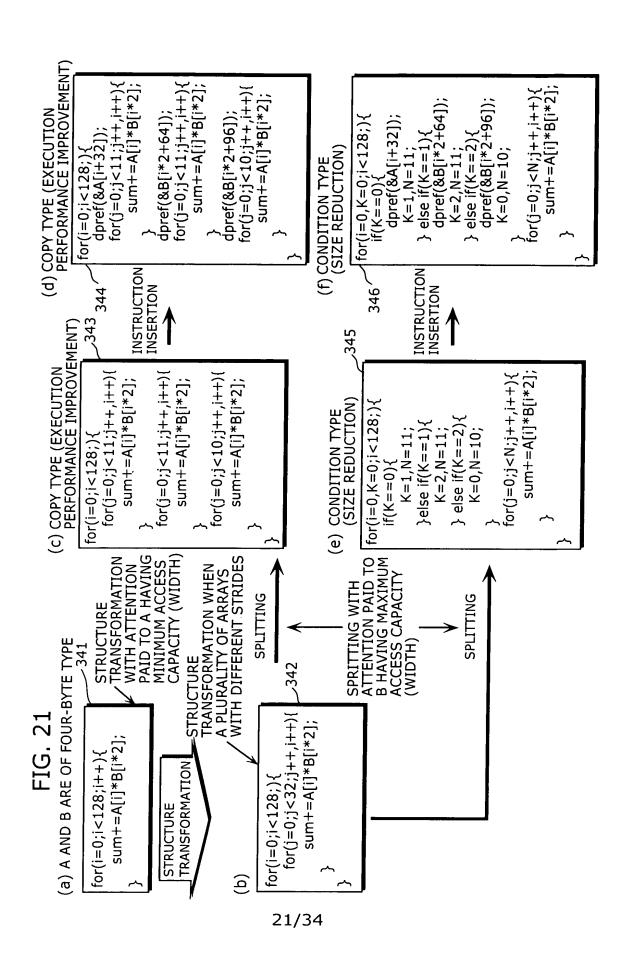


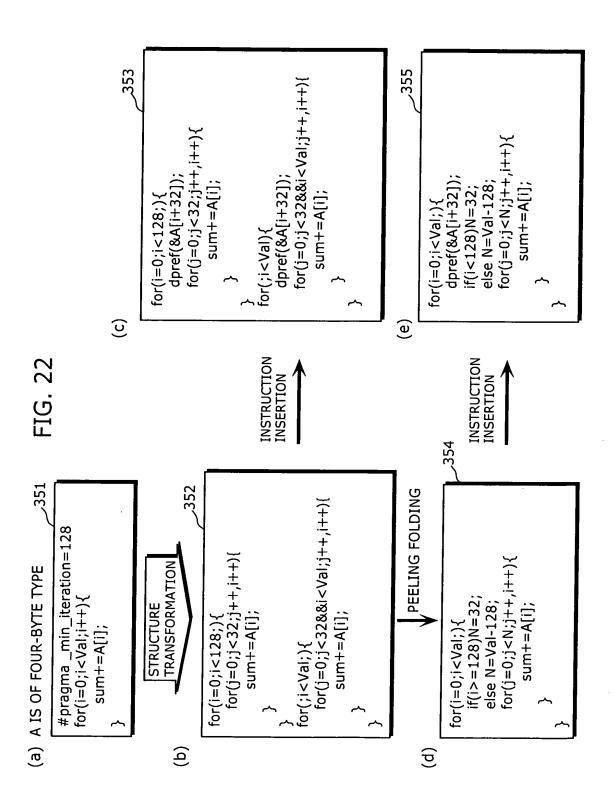












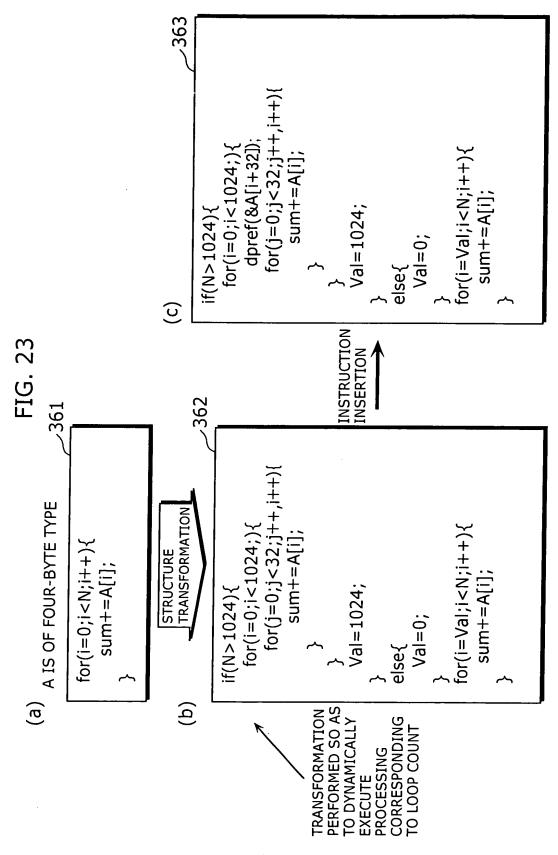


FIG. 24

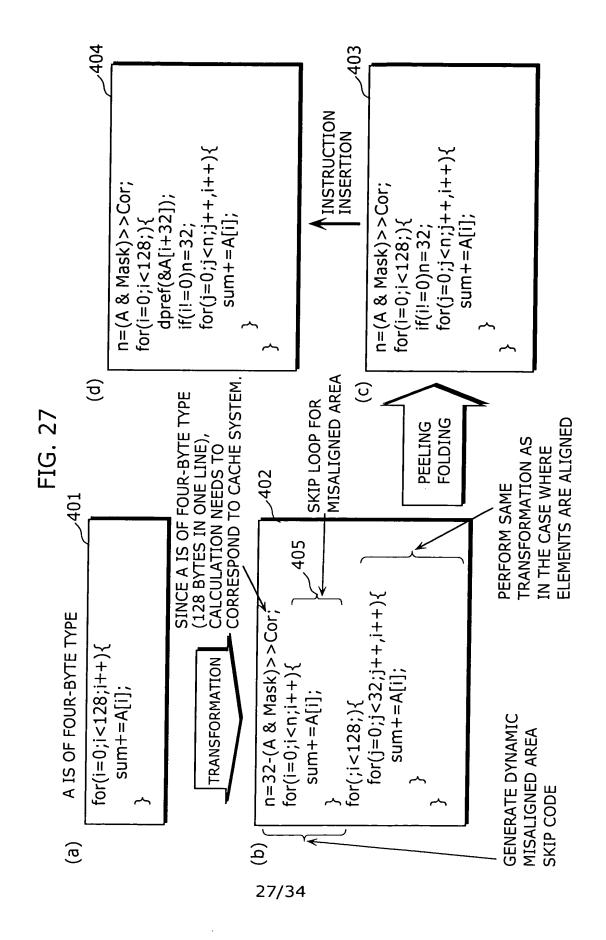
```
(a) A IS OF FOUR-BYTE TYPE
                                  J371
   for(i=0;i<N;i++){}
                                     WHEN IT IS JUDGED
                                     THAT LOOP STRUCTURE
      sum+=A[i];
                                     TRANSFORMATION
      sum+=A[i+1];
                                     IS UNNECESSARY,
      sum+=A[i+2];
                                     INSTRUCTION IS
        ~SKIP~
                                     INSERTED WITHOUT
      sum+=A[i+30];
                                     STRUCTURE
                                     TRANSFORMATION.
      sum+=A[i+31];
   }
                  INSTRUCTION
                  INSERTION
(b)
                                  -372
  for(i=0;i<N;i++){
      dpref(&A[i+32]);
     sum+=A[i];
     sum+=A[i+1];
     sum+=A[i+2];
        ~SKIP~
     sum+=A[i+30];
     sum+=A[i+31];
  }
```

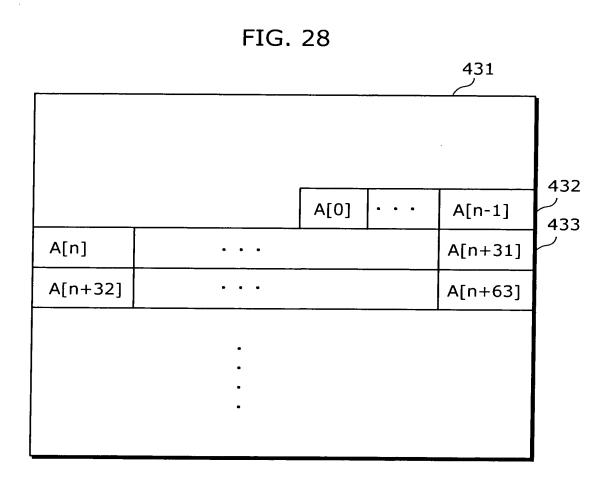
FIG. 25

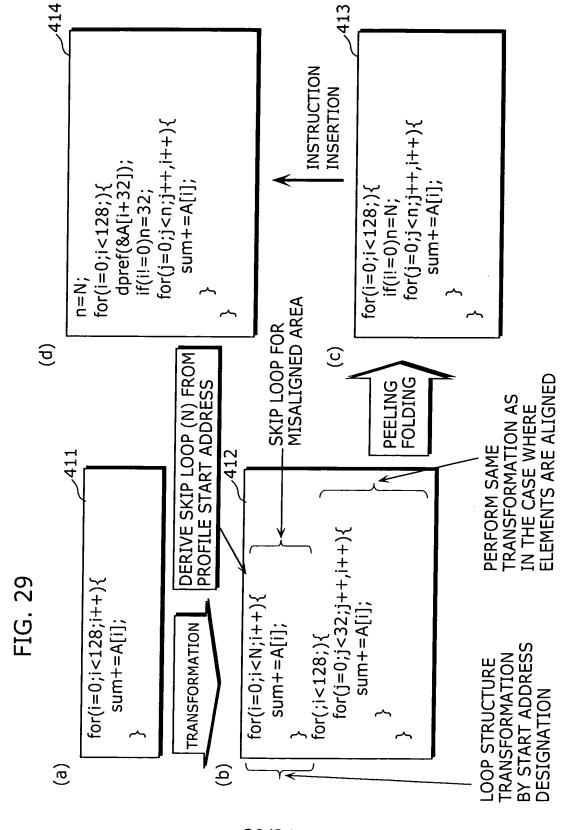
```
(a)
     A IS OF FOUR-BYTE TYPE
                                      381
     for(i=0;i<128;i++){
        sum+=A[i];
     }
                          INSTRUCTION INSERTION
                          (TWO LINES AHEAD)
                          AFTER NORMAL
       TRANSFORMATION
                          TRANSFORMATION
(b)
                                       SAME TRANSFORMATION
                                        AS IN THE CASE WHERE
     dpref(&A[0]);
                                        ELEMENTS ARE ALIGNED
     dpref(&A[32]);
     for(i=0;i<128;){
                                        382
       dpref(&A[i+64]);
       for(j=0;j<32;j++,i++){
          sum+=A[i];
       }
     }
```

FIG. 26

```
(a)
                                     391
    for(i=0;i<140;i++){
       sum+=A[i];
    }
                          INSTRUCTION INSERTION
                          (TWO LINES AHEAD)
                          AFTER NORMAL
       TRANSFORMATION
                          TRANSFORMATION
(b)
                                        SAME TRANSFORMATION
                                        AS IN THE CASE WHERE
    dpref(&A[0]);
                                        ELEMENTS ARE ALIGNED
    dpref(&A[32]);
                                        -392
   for(i=0;i<140;){
      dpref(&A[i+64]);
      if(i > = 128)n = 140-128;
      else n=32;
     for(j=0;j< n;j++,i++){
        sum+=A[i];
      }
   }
```







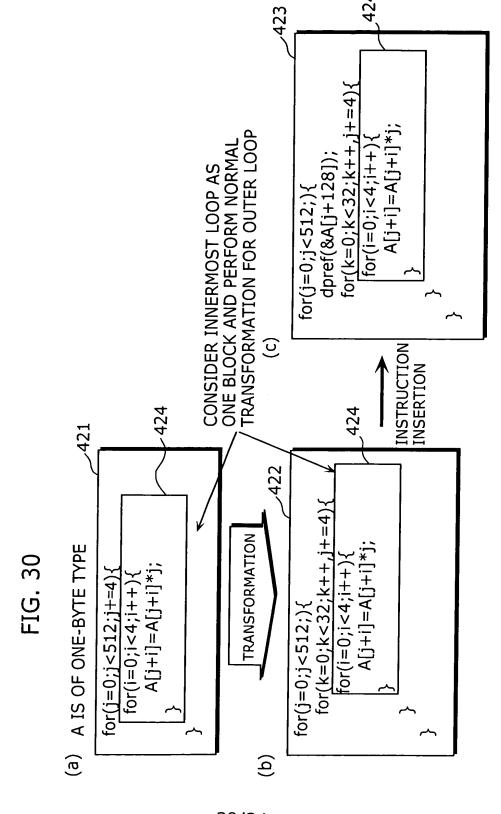


FIG. 31

```
int b[128]:
    #pragma _loop_tiling_dpref b
    for (i=0; i<128; i++)
    {
        a[i] = b[i];
    }</pre>
```

FIG. 32

```
(a)
       A IS OF FOUR-BYTE TYPE
                                   502
      for(i=0;i<128;i++) {
          A[i] = val * i;
            STRUCTURE
            TRANSFORMATION
(b)
                                   504
      for(i=0;i<128;) {
          for(j=0;j<32;j++,i++) {
             A[i] = val * i;
          }
             INSTRUCTION
              INSERTION
                                   √506
(c)
      for(i=0;i<128; ) {
          PreTouch(&A[i]);
          for(j=0;j<32;j++,i++) {
             A[i] = val * i;
```

